Zastosowania procesorów sygnałowych – projekt 2020

Projekt 2: Filtry FIR

Autor: Grzegorz Szwoch, greg@multimed.org, Katedra Systemów Multimedialnych, WETI PG

Aktualizacja: 07.05.2020

Wstęp

Celem zadania jest implementacja własnego filtru cyfrowego o skończonej odpowiedzi impulsowej (FIR) i przebadanie go za pomocą sygnałów testowych, wykorzystując wiedzę nabytą podczas realizacji poprzedniego zadania projektowego, a także wykorzystanie funkcji filtracji FIR z biblioteki DSPLIB i porównanie obu metod.

Zadaniem studentów jest:

- wykonanie poleceń opisanych w kolejnych punktach instrukcji,
- opracowanie uzyskanych wyników w formie raportu (każdy punkt instrukcji wymienia co powinno znaleźć się w raporcie, ponadto zaleca się zamieszczenie w raporcie własnych wniosków),
- umieszczenie w serwisie sprawozdań skompresowanego pliku, zawierającego: raport w formie pliku PDF oraz opracowany kod (plik *main.c*, ew. inne pliki dodane lub zmodyfikowane przez studenta); kod powinien być oczyszczony z niepotrzebnych rzeczy, czytelnie sformatowany i opatrzony komentarzami.

Materiały pomocne w wykonywaniu zadania to prezentacje z wykładów ZPS: *Filtry cyfrowe* oraz *Systemy liczbowe*, prezentacja *Filtry FIR* z wykładu *Przetwarzanie dźwięku i obrazu*, a także dokumentacja biblioteki DSPLIB (dostępna w materiałach pomocniczych na stronie Katedry oraz po wpisaniu SPRU422J w przeglądarce internetowej).

Przed rozpoczęciem wykonywania zadań należy wykonać w CCS kopię szablonu projektu i nazwać ją *FIR*. Wszystkie polecenia należy wykonać w tym projekcie.

1. Obliczenie współczynników filtru

Zadania:

- Obliczyć współczynniki dolnoprzepustowego filtru FIR o długości i częstotliwości granicznej podanej w osobnym dokumencie. Wykorzystać oprogramowanie do projektowania filtrów (Matlab, Python, itp.). Przyjąć częstotliwość próbkowania 48 kHz.
- Przekształcić współczynniki filtru do formatu Q15.
- Wykonać wykres charakterystyki częstotliwościowej filtru (dla stałoprzecinkowych współczynników).

- Sprawdzić stałe wzmocnienie filtru (sumę współczynników).
- Zapisać współczynniki w kodzie programu w pliku main.c, jako tablicę typu *const short*. Uwaga: nie umieszczać wszystkich współczynników w jednej linii. Długość linii kodu nie powinna przekraczać 120 znaków.

W raporcie:

- Zamieścić parametry projektowe filtru.
- Zamieścić wykres charakterystyki częstotliwościowej filtru. Ocenić, czy spełnia ona wymagania projektowe.
- Podać wartość stałego wzmocnienia filtru. Na tej podstawie ocenić ryzyko przepełnienia zakresu przy filtracji.

Podpowiedzi

Do obliczenia współczynników filtru należy użyć oprogramowania. Kilka z dostępnych możliwości:

- Matlab z *Signal Processing Toolbox* – jeżeli student ma dostęp do programu, może skorzystać z wygodnego narzędzia *fdatool*; dla studentów jest dostępna ograniczona czasowo licencja (2020) pod adresem: https://www.mathworks.com/licensecenter/classroom/COVID-19_Access/;
- Python z modułami *NumPy* i *SciPy* (oraz *matplotlib* do wykonania wykresów) – projektowanie filtru wymaga napisania skryptu, przykłady podano w prezentacji z wykładu PDiO;
- GNU Octave z modułem *Signal* – składnia kompatybilna z Matlabem, ale brak graficznego narzędzia, trzeba napisać skrypt.

Obliczone zmiennoprzecinkowe współczynniki filtru należy przekształcić do stałoprzecinkowego formatu Q15, z zaokrągleniem. Suma współczynników (stałe wzmocnienie) nie powinna przekraczać zakresu.

Charakterystykę częstotliwościową filtru wykreśla się jako moduł widma charakterystyki, w skali decybelowej. Wykres należy wykonać dla współczynników w formacie Q15. W Matlabie i Pythonie można użyć funkcji *freqz* do obliczenia charakterystyki widmowej.

2. Własna implementacja filtru FIR

Należy utworzyć funkcję o nazwie *blockfir*, która implementuje filtrację FIR za pomocą kodu języka C, nie korzystając z DSPLIB. Nagłówek funkcji powinien być następujący:

```
void blockfir(short* input, const short* filter, short* output, int numSamples,
int numFilter)
```

argumenty funkcji:

- *input*: wskaźnik do tablicy zawierającej próbki sygnału do przefiltrowania,
- *filter*: wskaźnik do tablicy zawierającej współczynniki filtru,
- *output*: wskaźnik do tablicy, w której zostaną zapisane wyniki filtracji,

- *numSamples*: liczba próbek w tablicy,
- *numFilter*: liczba współczynników filtru.

Zadania:

- Napisać funkcję *blockfir* wykonującą filtrację bloku próbek za pomocą instrukcji języka C (bez DSPLIB). Dla uproszczenia założyć, że funkcja będzie wywołana tylko raz (nie trzeba zapisywać stanu filtru, nie trzeba tworzyć bufora kołowego) oraz pominąć symetrię współczynników. Przy obliczaniu wyniku należy wykorzystać instrukcję MAC, poprzez funkcję *_smaci.*
- Zadeklarować dwie globalne tablice typu *short* o rozmiarze 5000 każda.
- Wypełnić jedną z tablic próbkami białego szumu.
- W funkcji *main* wywołać funkcję *blockfir* i zapisać wynik do drugiej tablicy. Po zatrzymaniu programu, wykonać i zapisać wykres widma przefiltrowanego sygnału (za pomocą CCS). Wykres powinien być wykonany w logarytmicznej skali amplitudy. Należy pominąć początkowe próbki, podając jako *Start address* np. *output + 200*, gdzie *output* jest nazwą tablicy zawierającej wyniki filtracji. Ocenić poprawność filtracji, ew. poprawić kod, jeżeli nie ma efektu filtracji dolnoprzepustowej.
- Wypełnić tablicę wejściową próbkami sygnału piłokształtnego o częstotliwości 100 Hz. Wykonać wykres widma sygnału (przed filtracją) w skali logarytmicznej oraz wykres czasowy. Zapisać wykresy.
- Wykonać filtrację sygnału piłokształtnego. Wykonać i zapisać wykres widma (w skali logarytmicznej, pomijając początkowe próbki) oraz wykres czasowy po filtracji (cała tablica).
- Zmierzyć i zanotować liczbę cykli zużywaną przez funkcję *blockfir*, używając narzędzi CCS, tak jak w poprzednim zadaniu projektowym (*Run* > *Clock* > *Enable*).

W raporcie:

- Zamieścić kod funkcji *blockfir*.
- Zamieścić wykres widma uzyskany dla filtracji białego szumu. Ocenić dokładność filtracji.
- Zamieścić i porównać wykresy widmowe dla sygnału piłokształtnego, przed i po filtracji. Opisać wpływ filtracji na widmo sygnału, zweryfikować poprawność filtracji.
- Zamieścić i porównać wykresy czasowe dla sygnału piłokształtnego, przed i po filtracji. Opisać wpływ filtracji na kształt sygnału. Wytłumaczyć, dlaczego początkowe próbki sygnału po filtracji są zniekształcone.
- Wyjaśnić, dlaczego używamy instrukcji MAC, zamiast po prostu napisać $y = y + a * x$.
- Wyjaśnić co się stanie, jeżeli jako pierwszy i trzeci argument funkcji podamy wskaźnik do tej samej tablicy.

Podpowiedzi

Mamy tablicę zawierającą wartości próbek i drugą tablicę zawierającą współczynniki filtru. Obie tablice zawierają liczby Q15. Naszym zadaniem jest przefiltrowanie każdej próbki i zapisanie wyniku do

osobnej tablicy wyjściowej. Potrzebne będą dwie pętle *for*. Zewnętrzna pętla będzie iterować po próbkach z tablicy wejściowej i zapisywać wynik filtracji we właściwym miejscu tablicy wyjściowej. Wewnętrzna pętla realizuje obliczenia dla jednej próbki. Musi ona wykonać mnożenie próbek sygnału przez odpowiadające im współczynniki filtru i sumować wyniki. Należy pamiętać, że próbki w tablicy są zapisane w kolejności od najstarszej od najmłodszej. Dlatego musimy iterować tę tablicę od bieżącej próbki (wyznaczonej przez aktualny indeks zewnętrznej pętli) w kierunku początku tablicy. Jeżeli dojdziemy do początku tablicy próbek i nie ma więcej próbek do obliczeń, musimy przerwać wewnętrzną pętlę.

Funkcja *_smaci* realizuje operację MAC, czyli mnożenie + akumulacja. Nagłówek funkcji jest następujący:

```
long _smaci(long y, int a, int x)
```

Instrukcja `y = _smaci(y, a, x)` wykonuje operacje: $y = y + a * x$ i zapisuje wynik do zmiennej typu *long*. Jeżeli a i x są w formacie Q15, to y jest w formacie Q30, tak jak przy standardowym mnożeniu. Próbka zapisywana do tablicy wyjściowej ma być w formacie Q15, musimy więc zaokrąglić liczbę i wykonać przesunięcie bitowe.

Sposób generowania białego szumu oraz sygnału piłokształtnego był przedmiotem zadania projektowego nr 1.

3. Filtracja blokowa sygnału za pomocą funkcji z DSPLIB

Zadania:

- W dokumentacji biblioteki DSPLIB znaleźć funkcję, która umożliwi wykonanie filtracji tak jak w poprzednim punkcie.
- W funkcji *main*, zakomentować poprzednią funkcję i wywołać funkcję z DSPLIB, przetwarzając całą tablicę za pomocą jednego wywołania funkcji.
- Przeprowadzić filtrację białego szumu i sygnału piłokształtnego oraz zapisać wyniki, tak jak w poprzednim punkcie (wystarczą wykresy widmowe w skali logarytmicznej).
- Zmierzyć i zanotować liczbę cykli procesora zużytą na wywołanie funkcji z DSPLIB.

W raporcie:

- Zamieścić kod instrukcji wykonującej filtrację.
- Zamieścić wykresy widmowe wyników filtracji. Skomentować dokładność filtracji oraz określić, czy są widoczne różnice w porównaniu z własną implementacją filtru.

Podpowiedzi

Dokumentacja DSPLIB (w języku angielskim) znajduje się na stronie Katedry w materiałach pomocniczych do projektu. Można ją również znaleźć wpisując SPRU422J w wyszukiwarce internetowej. Przy wywoływaniu funkcji należy pamiętać o rzutowaniu wskaźników do tablic na typ (DATA*). Funkcja filtracji z DSPLIB wymaga od nas utworzenia i podania bufora roboczego.

4. Filtracja sygnału próbka po próbce za pomocą funkcji z DSPLIB

Zadania:

- W funkcji *main*, zakomentować poprzednią funkcję. Zamiast tego, napisać pętlę, która wykona filtrację sygnału z tablicy wejściowej metodą „próbka po próbce”, czyli wywołując funkcję z DSPLIB dla każdej próbki osobno.
- Przeprowadzić filtrację białego szumu i sygnału piłokształtnego oraz zapisać wyniki, tak jak w poprzednich punkcie (wystarczą wykresy widmowe w skali logarytmicznej).
- Zmierzyć i zanotować liczbę cykli procesora zużytą na wykonanie całej pętli filtracji (wszystkich próbek w tablicy).

W raporcie:

- Zamieścić kod pętli wykonującej filtrację.
- Sprawdzić, czy wyniki filtracji są zgodne z uzyskanymi w poprzednich punktach.
- Zestawić wyniki pomiarów liczby cykli procesora dla wszystkich zbadanych przypadków. Wyjaśnić i skomentować różnice i ich przyczyny.
- Wyjaśnić w jakich praktycznych przypadkach zalecane będzie zastosowanie metody przetwarzania „próbka po próbce” zamiast przetwarzania blokowego.

Podpowiedzi

Dodatkowe wyjaśnienia nie powinny być potrzebne. Musimy napisać pętlę, która iteruje po wszystkich próbkach i wywołuje funkcję filtracji dla każdej próbki osobno. Oczywiście, nie możemy modyfikować bufora roboczego podawanego jako argument funkcji (wykorzystuje ona bufor roboczy do zapisania stanu filtru).

Koniec zadania projektowego